# Collaborative Editing within the Pervasive Collaborative Computing Environment

**Marcia Perry**
Lawrence Berkeley National Laboratory
One Cyclotron Road
Berkeley, CA 94720 USA
+1 510 486 6786
MPerry@lbl.gov

**Deb Agarwal**
Lawrence Berkeley National Laboratory
One Cyclotron Road
Berkeley, CA 94720 USA
+1 510 486 7078
DAAgarwal@lbl.gov

**ABSTRACT**
Scientific collaborations are established for a wide variety of tasks for which several communication modes are necessary, including messaging, file-sharing, and collaborative editing. In this position paper, we describe our work on the Pervasive Collaborative Computing Environment (PCCE) which aims to facilitate scientific collaboration within widely distributed environments. The PCCE provides a persistent space in which collaborators can locate each other, exchange messages synchronously and asynchronously and archive conversations. Our current interest is in exploring research and development of shared editing systems with the goal of integrating this technology into the PCCE. We hope to inspire discussion of technology solutions for an integrated approach to synchronous and asynchronous communication and collaborative editing.

**Keywords**
Collaborative editing, file sharing, synchronous messaging, asynchronous communication

## INTRODUCTION

Collaborative opportunities within the scientific community span the gamut from highly structured and scheduled meetings to informal and spontaneous interactions. The goals of scientific collaborations vary widely and can include such tasks as analyzing data sets and results from high energy physics experiments, coordinating local or remote equipment control, tracking workflow of remote job submissions, developing and debugging source code, and co-authoring publications. Researchers may be located within the same facility or organization or distributed across multiple domains, and collaboration can take place from different host environments varying from the user's desktop at his or her work site to a laptop or borrowed computer while traveling or at a conference. With the emergence of grid computing environments[8], collaborations between widely distributed participants are becoming increasingly common. Communication modes also vary from globally-scoped meetings and presentations via videoconferencing to synchronous discussions via instant messaging to asynchronous email exchanges.

Sharing documents is essential and at times it is the primary focus of a collaboration, but the means by which this is accomplished can vary considerably. Sometimes it is sufficient to read other's publications by viewing or downloading documents from web sites, perhaps with a discussion held over the telephone or email. Other times several people need to edit a shared document that is forwarded from author to author or accessed from a central repository. However certain situations call for simultaneously editing or annotating a shared file in real time.

In order to facilitate a variety of tasks within scientific collaborations, we have been researching and developing the Pervasive Collaborative Computing Environment (PCCE), which aims to provide a flexible environment that supports continuous or ad hoc interactions between widely distributed collaborators[1, 2]. The PCCE targets daily tasks such as synchronous and asynchronous messaging, file sharing and collaborative editing within a persistent and secure space. Additional goals are to offer a cross-platform and easy to use system and leverage existing tools and technologies as much as possible. We have developed and deployed the LBNLSecureMessaging system as the presence and messaging component of the PCCE. Our next goal is to support shared editing capabilities to further facilitate scientific collaboration. The next sections will discuss the LBNLSecureMessaging system and suggest ways in which collaborative editing might be integrated into the PCCE. We hope to inspire discussion of solutions for an integrated approach to group discussion and synchronous and asynchronous group editing.

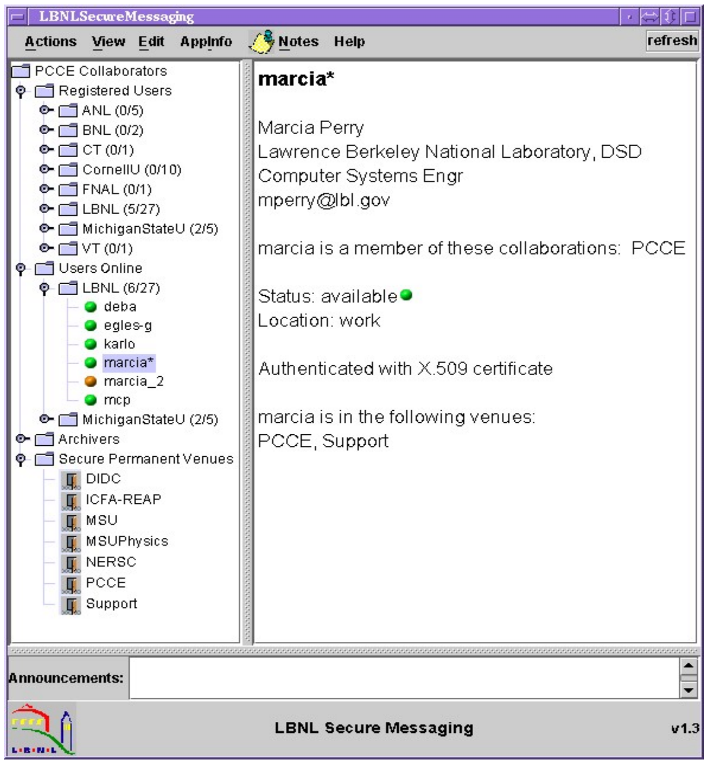
## THE PERVASIVE COLLABORATIVE COMPUTING ENVIRONMENT

The Pervasive Collaborative Computing Environment (PCCE) is being developed at LBNL as a flexible and seamless collaboration environment to support a continuum of collaborative interactions. It aims to provide a persistent space within which users can locate each other, exchange messages, share documents, and hold videoconferences. It is leveraging existing and recently proposed technologies and tools such as Internet Relay Chat [7], electronic notebooks, WebDAV[13], and whiteboard, video, and audio tools.

Within the PCCE, the LBNLSecureMessaging component has been developed and deployed as a secure presence and discussion environment to allow collaborators to locate each other, determine availability, and exchange messages synchronously or asynchronously. Within this system users can hold group or one-to-one conversations on an on-going or ad hoc basis. These conversations may be public and open to anyone who is on-line or they may be private and open only by invitation. All conversations take place within venues, which may be permanent or temporary. Permanent venues exist regardless of membership whereas temporary venues only exist while at least one person is present. Any venue may be made public or private by "opening the door" or "closing the door." To initiate a one-to-one conversation, users create a temporary private venue to which they invite someone else. The conversation can be extended to a larger group by its members' making the venue public or explicitly inviting others. Users may participate in multiple discussions simultaneously and they may leave notes for others who are on-line or off-line.

In order for users to easily locate each other and rendezvous, presence information is provided about people and venues. User information includes the person's name, nickname, email address, affiliation with one or more collaboration groups, and organization (including division and position within the organization). Additional information includes the user's location ("work," "home," or "mobile"), availability ("offline," "available," "busy," or "away"), and the venues of which he or she is a member. Users can sort their views of this information by different categories and they may dynamically create and maintain a list of "My Favorite People" that can be used to automatically accept or decline invitations, depending on availability. Venue information includes the name, mode (permanent or temporary, public or private), membership, and topic.

Figure 1 depicts the main window with presence information. Note that users may run multiple instances of themselves and the local instance is indicated by an asterisk added to the user's name. For example, "marcia" is the local user whose screen shot from her desktop at work is shown while "marcia_2" is at a conference but connected from her laptop computer. In color, the icons to the left of people's names are green, yellow, orange, and red to denote "available," "busy", "away," and "offline," respectively. Users may disconnect or change the status of remote instances of themselves from their local interface. The icon next to the "Notes" menu indicates there are notes which can be read, replied to, or deleted. Figure 2 shows a venue window in which a private conversation is taking place.



marcia*

Marcia Perry
Lawrence Berkeley National Laboratory, DSD
Computer Systems Engr
mperry@lbl.gov

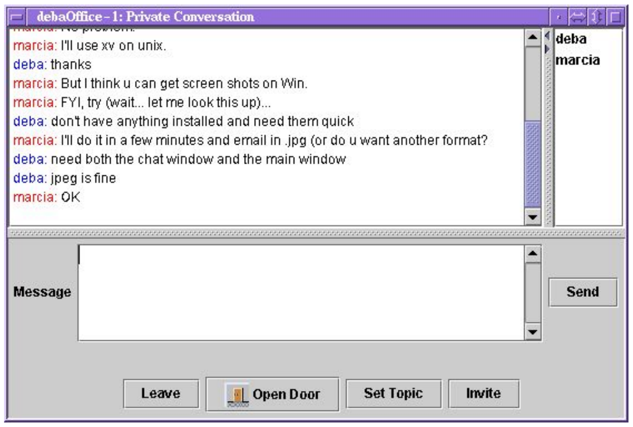
marcia is a member of these collaborations: PCCE

Status: available ●
Location: work

Authenticated with X.509 certificate

marcia is in the following venues:
PCCE, Support

**Figure 1: The LBNLSecureMessaging Main Window**



marcia: I'll use xv on unix.
deba: thanks
marcia: But I think u can get screen shots on Win.
marcia: FYI, try (wait... let me look this up)...
deba: don't have anything installed and need them quick
marcia: I'll do it in a few minutes and email in .jpg (or do u want another format?
deba: need both the chat window and the main window
deba: jpeg is fine
marcia: OK

**Figure 2: A Venue Window**

Our implementation follows a client-server model that supports authentication and encryption of messages exchanged over the network. We modified a public domain IRC server (IRCD hybrid) to replace its TCP sockets with SSL connections. We developed a custom PCCE server to provide persistence (e.g., unique nicknames and permanent venues) and enhanced presence information independent of any single chat server. To restrict access to known collaborators, pre-registration and login are required. The PCCE server stores registration information in a local database and uses this information to make authorization decisions (e.g., who can connect to the IRC server, who can leave and receive notes, and who can perform administrative operations).

Access to the environment is from a Java Swing interface that can be run continuously from the desktop for long-standing collaborations or launched for a short duration from a more mobile device such as a laptop computer. After having registered with the PCCE server (through either a system administrator or a registered user with administrative privileges), users log onto the system from the graphical user interface with either an X.509 credential or username and password. Users who authenticate with a certificate are granted extended capabilities (e.g., the ability to create new user accounts and permanent venues). Both the IRC and PCCE servers use only SSL connections and have their own X.509 certificates which are presented to each other and to clients.

The PCCE architecture is shown in Figure 3.

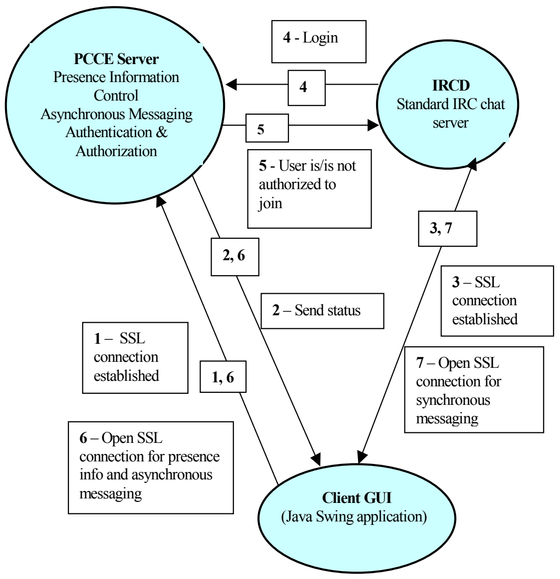

PCCE Server
Presence Information
Control
Asynchronous Messaging
Authentication &
Authorization
4 - Login
4
5
IRCD
Standard IRC chat server
5 - User is/is not authorized to join
3, 7
2, 6
3 – SSL connection established
2 – Send status
1 – SSL connection established
1, 6
7 – Open SSL connection for synchronous messaging
6 – Open SSL connection for presence info and asynchronous messaging
Client GUI
(Java Swing application)

**Figure 3: The PCCE Architecture**

The asynchronous messaging, presence information, remote control, and authorization services are components of the PCCE server (on the upper left). IRCD (on the upper right) implements the text-based synchronous chat messages using the IRC protocol. The client Java Swing graphical user interface is shown at the bottom as "Client GUI." A client starts the LBNLSecureMessaging interface and first establishes an SSL connection to the PCCE server to log into the authentication and authorization service, which verifies the validity of the identifying information against its database. If the login is successful, the PCCE server sends the current status including information about permanent venues, registered users, and availability of online users. The client then connects to the IRC server which queries the PCCE server about the user's authorization. If access to the chat facility is granted, all other users are sent the client's presence information and the client's connections to both servers are kept open to facilitate messaging and notification. All synchronous chat messages go directly to the IRC server, which forwards them to the targeted recipients through the established connections. The PCCE server sends notifications over established connections to the clients so that users can update their views of the collaboration group as other people join, change availability status, and leave.

## INTEGRATING COLLABORATIVE EDITING

Our vision is to extend the PCCE to support file-sharing and collaborative editing. The sort of scenario we would like to support, is illustrated in the following example. Recently we and two other people were writing a publication using a word processing program. Three authors were in the same building at LBNL (but in different offices and on different floors) and one author was in Virginia. We used the word processor's change tracking mechanisms to mark changes to the paper and the LBNLSecureMessaging tool to discuss and coordinate changes to the content of the paper.

After each author completed her changes, she would e-mail the changes to the rest of the group for review and possible integration. The next author would then review and accept or reject changes and he would begin the next round of changes. Using this methodology, the email became the versioning system and the change tracking became the means of comparing versions. This method of shared editing worked better than anything we have had in the past but still broke down at several points. For instance, there were several times when multiple authors wanted to work on the document concurrently. In these cases we negotiated a crude manual locking protocol which involved simply agreeing which sections each author was to edit. We also encountered cases where two authors simultaneously decided to work on the paper but did not inform each other of this fact. This often resulted in conflicting or redundant edits and one of the authors had to manually merge the two versions. In addition, the change tracking caused problems in the document unless each author accepted all the preceding changes.

If some mechanism for collaborative editing were integrated into the PCCE, we could have avoided the back-and-forth emailed attachments and confusion over whose turn it was to make or merge changes.

It is our experience that collaborative editing sessions often transition between synchronous and asynchronous interactions. Change tracking and a file checkout mechanism such as CVS can reasonably support purely asynchronous collaborative editing. But, synchronous or semi-synchronous editing requires a broader set of capabilities. Ideally a collaborative editing system would allow synchronous users a range of interaction capabilities during the editing process. These could include locking of sections of the document, a single unified view of the combined edits, ability to propose alternatives for a change, make comments on a change, randomly join and leave a collaborative editing session, and continue to edit the document during periods of disconnection from the network.

We could enhance our client interface to include controls for functions such as dropping documents into venues and opening them for reading and/or writing. The PCCE server's authorization service could be extended to control access to the shared files. For example, documents placed in public venues could be read-only while candidates for collaborative editing could be placed in private venues to which only authorized users would be granted write permission. Since the PCCE server now receives and forwards control messages it could be further utilized to interface with a shared editor.

Indeed several file-sharing and collaborative editing technologies and tools are available, such as Lotus Notes[6], WebEx[14], Zope[9], Wikis[15], CVW[4], Groove[10], BSCW[3], and Microsoft Word's support of document sharing via NetMeeting, to name just a few. However, many of these tools are proprietary, platform-specific, or tightly coupled with a specific chat tool. Others are centered around web site development or can only be accessed through a web browser. Although browser-based solutions offer ubiquity and platform independence, rich functioning, "heavy" clients may be more suitable as standalone GUI applications, especially with respect to response time and window management.

Web Distributed Authoring and Versioning (WebDAV) offers promise as a means of storing files in any format on a web server that basically acts as a centralized repository. Files can be uploaded from or downloaded to any platform and they can be locked and unlocked for synchronizing and merging document changes. Certainly the WebDAV directory tree on the web server can correspond to venues within the PCCE with security offered by means of HTTPS and file system access restrictions. However, file-locking insures taking turns and this restricts editing to be sequential.

In contrast to WebDAV and other CVS-like systems, replication of shared documents among multiple users allows update notification and concurrency control schemes that support real-time editing. Recently several alternatives for synchronous or real-time collaborative editing using replicated architectures have been described. The intelligent collaboration transparency (ICT) infrastructure is a means by which off-the-shelf single-user editors can be converted into groupware without source code modification. A prototype implementation has been developed for MSWord and Gvim under MS Windows [11]. Although many scientific researchers write publications under Unix, usually with Latex and Framemaker, many authors are using or switching to MS Word from Windows or StarOffice on Unix. Allowing simultaneous editing of Word documents with the ability to discuss them outside of NetMeeting would be beneficial.

Also under development is a reference implementation of an operational transformation strategy for concurrency control and consistence maintainance to enable synchronous collaborative editing of metadata-rich hierarchical content such as SGML documents[5]. Web service developers (such as authors of Grid services) who write and edit XML and HTML documents would probably benefit from this approach.

Notification is used extensively in collaborative applications to allow changes made by one user to propagate to others. For example, chat tools such as ICQ and LBNLSecureMessaging forward messages to targeted recipients immediately or shortly after they are composed, usually when the sender presses the return key or clicks a "Send" button. The Realtime Distributed Unconstrained Cooperative Editing (REDUCE) system uses a notification scheme that focuses on high responsiveness by allowing document updates to be propagated immediately and frequently. Such an editor would facilitate real-time editing and discussion. Document-sharing tools such as CVS make updates known to others when the user who has edited the file issues a specific command such as "commit," "update," or "publish." This is usually done infrequently and after a fairly lengthy editing session. This paradigm is useful in situations in which individual members of development teams are working on a shared document offline or over a long period of time.

However, we envision a need for both approaches. One such scenario involves group source code development in which one user has a file checked out of a repository and locked for a long period of time to work on one set of functions while another group wishes to edit and discuss different sections of the code simultaneously in real time. We recall a face-to-face meeting in which many physicists were defining requirements for building a collaboratory focused on a global accelerator. A mediator recorded, assessed, and summarized the ideas expressed by the participants. If the meeting had been online with the distributed members connected from their work sites, the content generated by the physicists would have needed to be propagated to the mediator and to each other automatically. The mediator could have transmitted her content as it was being written or in chunks after some delay (e.g., "publish" or "send" after each paragraph or after a list was compiled).

A flexible framework has been developed to support multiple notification strategies so that document updates can be propagated in both real-time and non-real-time. Based on this notification scheme is the Notification-flexIble Collaborative Editing (NICE) system [12] which is being implemented as a successor to REDUCE. This type of group editor would facilitate the combination of real-time and non-real-time editing depicted by the group code development and brain-storming scenarios cited above.

## CONCLUSION

We have described our development of the Pervasive Collaborative Computing Environment (PCCE) which aims at supporting continuous or ad hoc communication within scientific collaborations. We are attempting to leverage as much as possible from existing tools and infrastructures. Accordingly, we have implemented a prototype presence and messaging system, the LBNLSecureMessaging component of the PCCE, which is based on an existing IRC server. We have modified this server and enhanced it with persistence and security capabilities that were needed to fulfill our users' requirements. Our next goal is to integrate shared editing capabilities into the PCCE. Since our users work from different types of computers (desktops and laptops), from different operating systems (Unix and Windows), and from different parts of the globe, Java-based systems and those that support both real-time and non-real-time are very attractive. The intention of this paper is to explore and discuss solutions for an integrated approach to synchronus and asynchronous communication and document sharing and editing. While CVS-type strategies address the cross-platform issue and facilitate asynchronous or non-real-time editing, more recent developments in collaborative editing, based on replicated distributed paradigms, are attractive for real-time application sharing. We envision a hybrid scheme in which users can choose the group editing mode (automatic publishing of updates versus issuing specific 'send' commands periodically). We invite opportunities for incorporation of existing prototypes for collaborative editing into the PCCE environment.

**Acknowledgments**
This work was supported by the U.S. Department of Energy, Office of Science, Mathematical, Information and Computation Sciences under contract DE-AC03-76SF0098. This paper is also available as LBNL Report number LBNL-53769.

## REFERENCES

1. Agarwal, D., Lorch, M., Thompson, M., Perry, M. A New Security Model for Collaborative Environments, *Proceedings of the Workshop on Advanced Collaborative Environments*, Seattle,WA, June 2003. LBNL-52894.
2. Agarwal, D., McParland, P., Perry, M., Supporting Collaborative Computing and Interaction, *Proceedings of the Grace Hopper Celebration of Women in Computing 2002 Conference*, Vancouver, Canada, October 2002.
3. Appelt, W., What Groupware Functionality do Users Really Use?, *Proceedings of the 9th Euromicro Workshop on PDP 2001*, Mantua, February 2001, IEEE Computer Society, Los Alamitos.
4. Collaborative Virtual Workspace (CVW), http://cvw.sourceforge.net/.
5. Davis, A. H., Sun, C., Lu, J., Generalizing Operational Transformation to the Standard General Markup Language, *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW2002)*, New Orleans, LA, November 2002.
6. Falkner, M, *Using Lotus Notes as an Intranet*, John Wiley & Sons, March, 1997.
7. Internet Relay Chat (IRC):http://www.irc.org/.
8. Johnston, W., Vision for Collaboratories and the DOE Science Grid, *Presentation at the DOE2000 Review*, May 2000, Argonne, Illinois.
9. Lattier, A., Pelletier, M., *The Zope Book*, New Riders Publishing, 1st edition, July, 2001.
10. Levine, J., *10 Minute Guide to Groove 2.0*, Que Publisher, June 2002.
11. Li, D., Li, R., Transparent Sharing and Interoperation of Heterogenous Single-User Applications, *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW2002)*, New Orleans, LA, November 2002.
12. Shen, H., Sun, C., Flexible Notification for Collaborative Systems, *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW2002)*, New Orleans, LA, November 2002.
13. webdav: IETF WEBDAV Working Group, http://ftp.ics.uci.edu/pub/ietf/webdav.
14. WebEx, http://www.webex.com/.

15. WikiWikiWeb,
http://c2.com/cgi/wiki?WikiWikiWeb.